15-812: Programming Language Semantics

Lecture Notes on Games and Interaction

André Platzer

Carnegie Mellon University
Lecture 14

1 Introduction

The purpose of this lecture is to broaden our horizon of what programs can be good for and what interesting twists on semantical challenges that brings. We will move on from the study of classical sequential computation to proceed to the study of interactive computation. In sequential computation, one step of computation happens after the other but the program is entirely in charge of resolving each of these steps. In the context of a box modality $[\alpha]\phi$ we are interested in establishing that every which way program α runs, the formula ϕ is true afterwards. For a diamond modality $\langle\alpha\rangle\phi$ we want to show that there is one way of running program α such that formula ϕ is true afterwards. But there is only one mode of running α. Either all or just some of its computations are relevant, depending on the modality around it. These are good models if either every choice is under our control and works in our favor. Or if no choice is under our control and nothing works in our favor. In mixed cases, where some choices help and others hinder, this sequential computation is the wrong model.

In today's lecture, we will consider mixed cases of interactive computation in games where some of the choices during the execution of a program are being resolved in our favor, while others are not. This is a good model for fully adversarial situations such as in security where some actions are performed by benevolent parties while others could be performed by malevolent adversaries. It's also a good model for cases where two parties interact that may simply act against each other's interests out of ignorance. For example a command line shell needs to be able to react on some good way no matter what input a user provides, which is an example of game dynamics resulting from interaction. Real-time schedulers need to find some way of reacting no matter what tasks come in at what rates dynamically over time. Also, complexity theory has a good understanding of best-case complexity and worst-case complexity, which correspond

to properties of diamond and box modalities of programs respectively. It would take a game understanding to provide mixed-case complexity where some aspects of the computation are assumed to help others are assumed to be resolved against. More generally, games are not just a good model for true adversarial competition but can already arise for benign multi-agent situations in cases where they might accidentally compete because of a lack of permanent perfect coordination.

These lecture notes are based on [Pla15, Pla18].

2 Choices & Nondeterminism

Note 1 (Choices)**.** *Systems involve choices. They manifest evidently as nondeterministic choices $\alpha \cup \beta$ whether to run HP α or HP β and in nondeterministic repetitions α^* where the choice is how often to repeat α. All those choices, however, have still been resolved in one way, i.e. by the same entity or player.*

In particular, choices in α help $\langle\alpha\rangle\phi$, because what this formulas calls for is *some* way of making ϕ happen after α. If α has many possible behaviors, this is easier to satisfy. Choices in α hurt $[\alpha]\phi$, however, because this formula requires ϕ to hold for all those choices. The more choices there are, the more difficult it is to make sure that ϕ holds after every single combination of those choices.

Note 2. *In dynamic logic, choices in α either help uniformly (when they occur in $\langle\alpha\rangle\phi$) or make things more difficult uniformly (when they occur in $[\alpha]\phi$).*

That is why these various forms of choices in programs have been called *nondeterministic*. They are "unbiased". All possible resolutions of the choices in α could happen nondeterministically when running α. Which possibilities we care about (all or some) just depends on the modal formula around it.

3 Control & Dual Control

Another way of looking at the choices that are to be resolved during the runs of a program α is that they can be resolved by one player. Let's call her *Angel*, because she helps us so much in making $\langle\alpha\rangle\phi$ formulas true. Whenever a choice is about to happen (by running the program statements $\alpha \cup \beta$ or α^*, Angel is called upon to see how the choice is supposed to be resolved this time.

From that perspective, it sounds easy enough to add a second player. Let's call him *Demon* as Angel's perpetual opponent.[1] Only so far, Demon will probably be rather bored after a while, when he realizes that he never actually gets to decide anything,

[1] The responsibilities of such ontologically loaded names are easier to remember than those of neutral player names I and II.

because Angel has all the fun in choosing how the program world unfolds and Demon just sits around idly. So to keep Demon entertained, we need to introduce some choices that fall under Demon's control.

One thing, we could do to keep Demon interested in playing along is to add a pair of shiny new controls especially for him. They might be called $\alpha \cap \beta$ for Demon's choice between α or β as well as $\alpha^\times$ for repetition of α under Demon's control. But that would cause a lot of attention to Demon's control, which might make him feel overly majestic. Let's not do that, because we don't want Demon to get any ideas.

Instead, we will find it sufficient to add just a single operator to programs: the dual operator ^{d}. What α^d does is to give all control that Angel had in α to Demon, and, vice versa, all control that Demon had in α to Angel. The dual operator, thus, is a little bit like what happens when you turn a chessboard around by $180°$ in the middle of the game. Whoever played the choices of player White before suddenly controls Black, and whoever played Black now controls White. With just this single duality operator it turns out that Demon still gets his own set of controls ($\alpha \cap \beta$ and $\alpha^\times$ by suitably nesting the operators, but we did not have to give him those controls specifically. Yet, now those extra controls are not special but simply an aspect of a more fundamental principle: duality.

4 Games

Differential game logic (dGL) is a logic for studying properties of hybrid games [Pla15] of which we will consider the special case where everything is entirely discrete. The idea is to describe the game form, i.e. rules, dynamics, and choices of the particular game of interest, using a program notation and to then study its properties by proving the validity of logical formulas that refer to the existence of winning strategies for objectives of those games.

Definition 1 (Games)**.** The *games of game logic* are defined by the following grammar (α, β are games, x a variable, θ a term, Q is a first-order logic formula):

$$\alpha, \beta \ ::= \ x := \theta \mid ?Q \mid \alpha \cup \beta \mid \alpha; \beta \mid \alpha^* \mid \boldsymbol{\alpha^d}$$

The *atomic games* are assignments, continuous evolutions, and tests. In the *deterministic assignment game* (or discrete assignment game) $x := \theta$, the value of variable x changes instantly and deterministically to that of θ by a discrete jump without any choices to resolve. The *test game* or *challenge* $?Q$ has no effect on the state, except that Angel loses the game immediately if formula Q does not hold in the current state, because she failed the test she was supposed to pass. The test game $?Q$ challenges Angel and she loses immediately if she fails. Angel does not win just because she passed the challenge $?Q$, but at least the game continues. So passing challenges is a necessary condition to win games. Failing challenges, instead, immediately makes Angel lose.

The *compound games* are sequential, choice, repetition, and duals. The *sequential game*

$\alpha;\beta$ is the game that first plays game α and, when game α terminates without a player having won already (so no challenge in α failed), continues by playing game β. When playing the *choice game* $\alpha \cup \beta$, Angel chooses whether to play game α or play game β. Like all the other choices, this choice is dynamic, i.e. every time $\alpha \cup \beta$ is played, Angel gets to choose again whether she wants to play α or β this time. The *repeated game* α^* plays game α repeatedly and Angel chooses, after each play of α that terminates without a player having won already, whether to play the game again or not, albeit she cannot choose to play indefinitely but has to stop repeating ultimately. Angel is also allowed to stop α^* right away after zero iterations of α. Most importantly, the *dual game* α^d is the same as playing the game α with the roles of the players swapped. That is Demon decides all choices in α^d that Angel has in α, and Angel decides all choices in α^d that Demon has in α. Players who are supposed to move but deadlock lose. Thus, while the test game $?Q$ causes Angel to lose if formula Q does not hold, the *dual test game* (or *dual challenge*) $(?Q)^d$ instead causes Demon to lose if Q does not hold. For example, if α describes the game of chess, then α^d is chess where the players switch sides.

The presence of ^{d} requires a thorough semantic generalization the logic to cope with such flexibility.

5 Game Logic

Games describe how the world can unfold when Angel and Demon interact according to their respective control choices. They explain the rules of the game how Angel and Demon interact, but not who wins the game, nor what the respective objectives of the players are.[2] The winning conditions are specified by logical formulas of game logic. Modal formulas $\langle\alpha\rangle\phi$ and $[\alpha]\phi$ refer to games and the existence of winning strategies for Angel and Demon, respectively, in a game α with a winning condition specified by a logical formula ϕ.

Definition 2 (Game logic formulas)**.** The *formulas of game logic* are defined by the following grammar (ϕ, ψ are formulas, p is a predicate symbol of arity k, θ_i are (polynomial) terms, x a variable, and α is a game):

$$\phi, \psi \ ::= \ p(\theta_1, \ldots, \theta_k) \mid \theta_1 \geq \theta_2 \mid \neg\phi \mid \phi \wedge \psi \mid \exists x\, \phi \mid \langle\alpha\rangle\phi \mid [\alpha]\phi$$

Other operators $>, =, \leq, <, \vee, \rightarrow, \leftrightarrow, \forall x$ can be defined as usual, e.g., $\forall x\, \phi \equiv \neg\exists x\, \neg\phi$. The modal formula $\langle\alpha\rangle\phi$ expresses that Angel has a winning strategy to achieve ϕ in game α, i.e. Angel has a strategy to reach any of the states satisfying formula ϕ when playing game α, no matter what strategy Demon chooses. The modal formula $[\alpha]\phi$ expresses that Demon has a winning strategy to achieve ϕ in game α, i.e. a strategy to

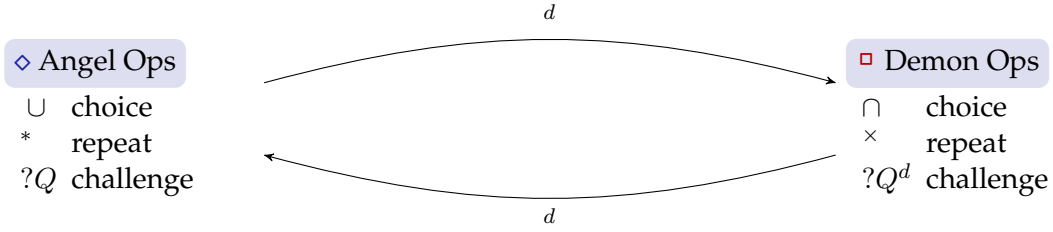
[2] Except that players lose if they disobey the rules of the game by failing their respective challenges.

reach any of the states satisfying ϕ, no matter what strategy Angel chooses.[3] Note that the same game is played in $[\alpha]\phi$ as in $\langle\alpha\rangle\phi$ with the same choices resolved by the same players. The difference between both formulas is the player whose winning strategy they refer to. Both use the set of states where formula ϕ is true as the winning states for that player. The winning condition is defined by the modal formula, α only defines the game form, not when the game is won, which is what ϕ does. game α defines the rules of the game, including conditions on state variables that, if violated, cause the present player to lose for violation of the rules of the game. The formulas $\langle\alpha\rangle\phi$ and $[\alpha]\neg\phi$ consider complementary winning conditions for Angel and Demon.

6 Demon's Controls

Angel has full control over all choices in each of the operators of games *except* when the operator ^{d} comes into play. All choices within the scope of (an odd number of) ^{d} belong to Demon, because ^{d} makes the players switch sides. Demon's controls, i.e. direct controls for Demon, can be defined using the duality operator ^{d} on Angel's controls.

Demonic choice between game α and β is $\alpha \cap \beta$, defined by $(\alpha^d \cup \beta^d)^d$, in which either the game α or the game β is played, by Demon's choice. The choice for the $\cup$ operator belongs to Angel, yet since it is nested within ^{d}, that choice goes to Demon, except that the ^{d} operators around α and β restore the original ownership of controls. *Demonic repetition* of game α is $\alpha^\times$, defined by $((\alpha^d)^*)^d$, in which α is repeated as often as Demon chooses to. Again, the choice in the * operator belongs to Angel, but in a ^{d} context goes to Demon, while the choices in the α, β subgames underneath stay as they were originally thanks to the additional ^{d} operators. In $\alpha^\times$, Demon chooses after each play of α whether to repeat the game, but cannot play indefinitely so he has to stop repeating ultimately. Dual assignment $(x := \theta)^d$ is equivalent to $x := \theta$, because it never involved any choices to begin with. Angel's control operators and Demon's control operators correspond to each other by duality:



7 Examples

A few examples of games are as follows, some of which are valid and some are only under certain circumstances:

[3] It is easy to remember which modal operator is which. The formula $\langle\alpha\rangle\phi$ clearly refers to Angel's winning strategies because the diamond operator $\langle\cdot\rangle$ has wings.

$$\vDash\langle(x := x+1; (x := x+1)^\times \cup x := x-1)^*\rangle\, x = 0$$
$$\nvDash\langle(x := x+1; (x := x+1)^\times \cup (x := x-1 \cap x := x-2))^*\rangle\, x = 0$$

$$\Big\langle((w := b \cap w := -b \cap w := 0);$$
$$(v := a \cup v := -a \cup v := 0);$$
$$(x := x+v; y := y+w)^*\Big\rangle\, (x-y)^2 \leq 10$$

$$\Big\langle((v := a \cup v := -a \cup v := 0);$$
$$(w := b \cap w := -b \cap w := 0);$$
$$(x := x+v; y := y+w)^*\Big\rangle\, (x-y)^2 \leq 10$$

Balanced scheduling game with j jobs with p scheduled on CPU1 and q being scheduled on CPU2:

$$\vDash\Big[j := 0; p := 0; q := 0;$$
$$\big((n := 0 \cup n := 1 \cup n := 2); \quad // \text{ start } n \text{ jobs}$$
$$(p := p+1 \cap ?true); (q := q+1 \cap ?true); \quad // \text{ may schedule jobs on either CPU}$$
$$j := j+n$$
$$\big)^*\Big]\, (|p-q| \leq 1 \wedge p+q = j)$$

The scheduler played by Demon needs to make sure the jobs are balanced with at most 1 imbalance between both CPUs while also making sure to schedule all jobs without overscheduling too many jobs.

8 Operational Game Semantics (informally)

Treatment of a proper semantics for game logic will be deferred to the next lecture. A graphical illustration of the choices when playing games is depicted in Fig. 1. The nodes where Angel gets to decide are shown as diamonds ◇, the nodes where Demon decides are shown as boxes □. Circle nodes are shown when it depends on the remaining game which player it is that gets to decide. Dashed edges indicate Angel's actions, solid edges would indicate Demon's actions, while zigzag edges indicate that a game is played and the respective players move as specified by that game. The actions are the choice of playing the left or the right game for a choice game $\alpha \cup \beta$, and the choice of whether to stop or repeat in a repeated game α^*. This principle can be made rigorous in an operational game semantics [Pla15], which conveys the intuition of interactive game

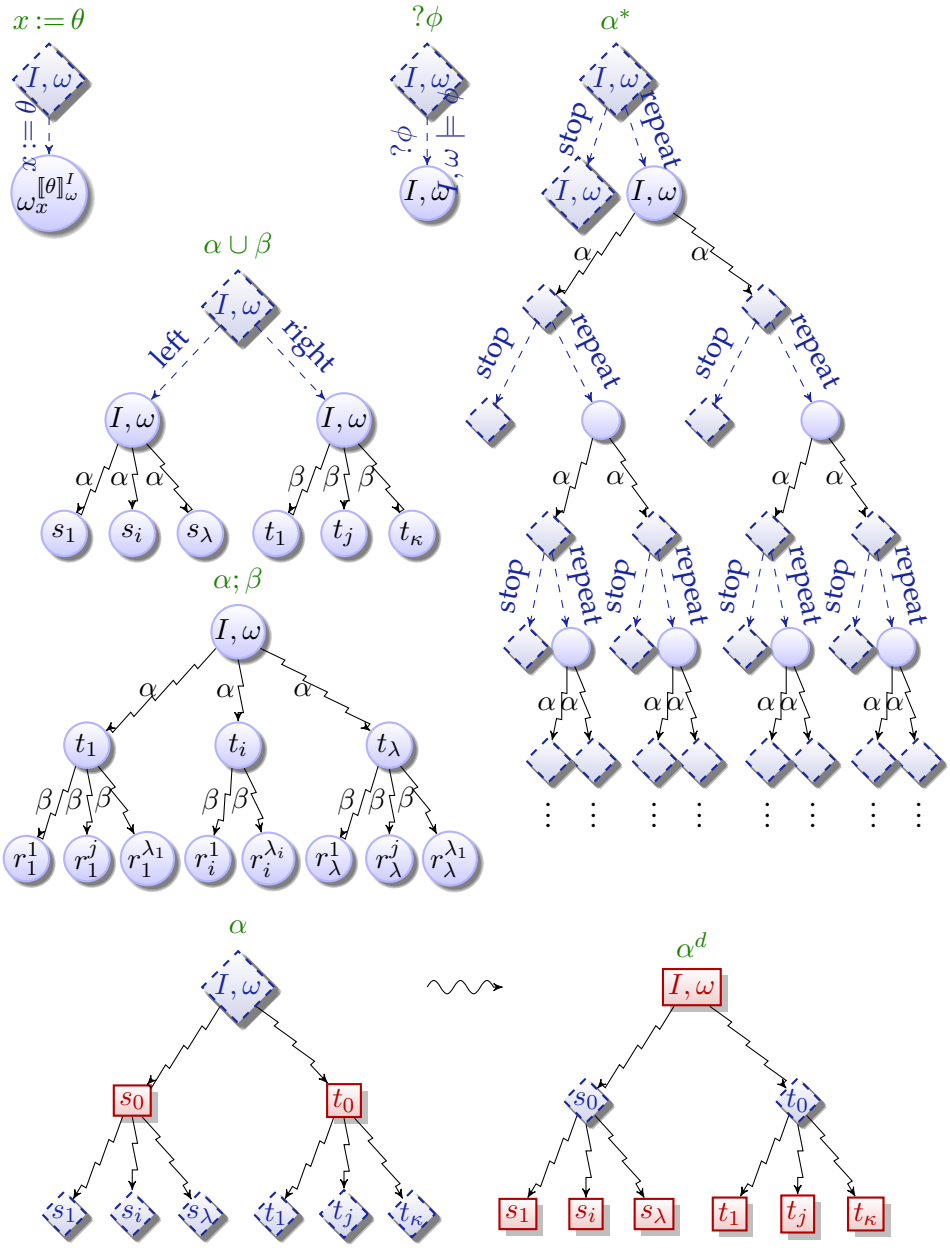


Figure 1: Operational game semantics

play for games, relates to game theory and descriptive set theory, but is also beyond the scope of these lecture notes. Observe how all choices involve at most two possibilities.

As an example, consider the *filibuster formula*:

$$\langle (x := 0 \cap x := 1)^* \rangle x = 0 \tag{1}$$

It is Angel's choice whether to repeat (*), but every time Angel repeats, it is Demon's choice ($\cap$) whether to play $x := 0$ or $x := 1$. What is the truth-value of the formula (1)?

The game in this formula never deadlocks, because every player always has a remaining move (here even two). But it may appear as if the game had perpetual checks, because no strategy helps either player win the game; see Fig. 2. How could that happen and what can be done about it?

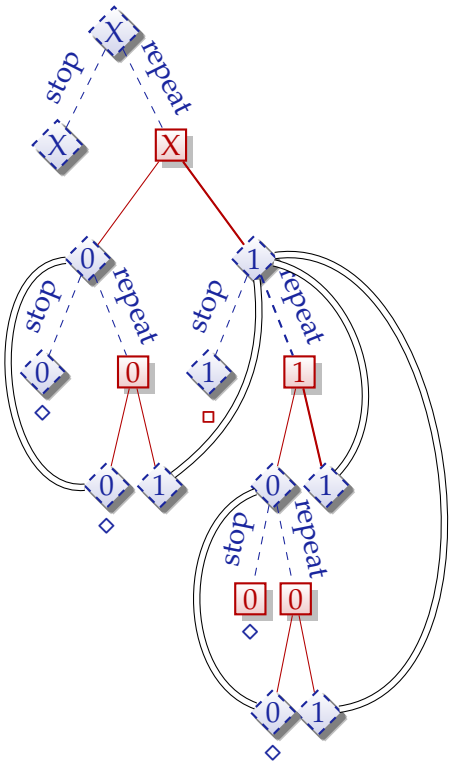


Figure 2: The filibuster game formula $\langle (x := 0 \cap x := 1)^* \rangle x = 0$ looks like it might be non-determined and not have a truth-value (unless $x = 0$ initially) when the strategies follow the thick actions. Angel's action choices are illustrated by dashed edges from dashed diamonds, Demon's action choices by solid edges from solid squares, and double lines indicate identical states with the same continuous state and a subgame of the same structure of subsequent choices. States where Angel wins are marked ◇ and states where Demon wins by ▫.

Before you read on, see if you can find the answer for yourself.

The mystery of the filibuster game can be solved when we remember that the game still ultimately ought to stop in order to be able to inspect who won the game. Angel is in charge of the * repetition and she can decide whether to stop or repeat. The filibuster game has no tests, so the winner only depends on the final state of the game, because both players are allowed to play arbitrarily without having to pass tests in between. Angel wins a game play if $x = 0$ holds in the final state and Demon wins if $x \neq 0$ holds in the final state. What do the strategies indicated in Fig. 2 do? They postpone the end of the game forever, hence there would never be a final state in which it could be evaluated who won. That is, indeed, not a way for anybody to win anything. Yet, Angel was in charge of the repetition *, so it is really her fault if the game never comes to a stop to evaluate who won, because she has to call it quits at some point. Consequently, the semantics of games requires players to repeat as often as they want but they cannot repeat indefinitely. This will be apparent in the actual semantics of games, which is defined as a denotational semantics corresponding to winning regions. Thus, (1) is false unless $x = 0$ already holds initially.

The same phenomenon happens in

$$\langle (x := 0; (x := x + 1)^\times)^* \rangle x = 0 \tag{2}$$

in which both players can let the other one win. Demon can let Angel win by choosing to evolve for duration 0. And Angel can let Demon win by choosing to stop even if $x \neq 0$. Only because Angel will ultimately have to stop repeating does the formula in (2) have a truth-value and the formula is false unless $x = 0$ already holds initially.

9 Summary

This lecture saw the introduction of game logic, which extends dynamic logic with capabilities of modeling and understanding games. Compared to systems, the new dynamical aspect of adversarial dynamics is captured entirely by the duality operator ^{d}. Without it, games are single-player games, which are equivalent to systems.

After this lecture showed an informal and intuitive discussion of the actions that games allow, the next lecture gives a proper semantics to game logic and their games.

Exercises

Exercise 1. Single player games, i.e. ^{d}-free games, are just programs. For each of the following formulas, convince yourself that it has the same meaning, whether you understand it as a dynamic logic formula with a systems or as a game logic formula with

a game (that happens to have only a single player):

$$\langle x := 0 \cup x := 1\rangle x = 0$$
$$[x := 0 \cup x := 1]x = 0$$
$$\langle (x := 0 \cup x := 1); ?x = 1\rangle x = 0$$
$$[(x := 0 \cup x := 1); ?x = 1]x = 0$$
$$\langle (x := 0 \cup x := 1); ?x = 0\rangle x = 0$$
$$[(x := 0 \cup x := 1); ?x = 0]x = 0$$
$$\langle (x := 0 \cup x := 1)^*\rangle x = 0$$
$$[(x := 0 \cup x := 1)^*]x = 0$$
$$\langle (x := 0 \cup x := x + 1)^*\rangle x = 0$$
$$[(x := 0 \cup x := x + 1)^*]x = 0$$

References

[Pla15] André Platzer. Differential game logic. *ACM Trans. Comput. Log.*, 17(1):1:1–1:51, 2015. doi:10.1145/2817824.

[Pla18] André Platzer. *Logical Foundations of Cyber-Physical Systems*. Springer, Switzerland, 2018. URL: http://www.springer.com/978-3-319-63587-3.